```
1.  **5,649,095**, Jul. 15, 1997, Method and apparatus for detecting
computer viruses through the use of a scan information cache; Paul D.
Cozza, 395/183.15; 380/4; 395/440 [IMAGE AVAILABLE]
=> display clms
ENTER (L6), L# OR ?:16
ENTER ANSWER NUMBER OR RANGE (1):1

US PAT NO:     **5,649,095** [IMAGE AVAILABLE]          L6: 1 of 1

CLAIMS:

CLMS(1)

 What is claimed is:

 1. A method for increasing the speed at which a computer, which has
files including more than one fork, scans for the presence of a computer
virus, said method comprising the steps of:
 creating a scan information cache on a non-volatile storage medium;
 gathering identifying information, which includes at least one length of
  some portion of a file, about an initial state of said file;
 storing said identifying information in said scan information cache;
 gathering current state information, which includes at least one length
  of some portion of said file, about a current state of said file;
 determining how said identifying information stored in said scan
  information cache differs from said current state information thereby
  indicating a presence or absence of one or more subsets of computer
  viruses, said one or more subsets each including one or more viruses
  which affect state information of said file in certain characteristic
  manners;
 scanning said file for one or more of said subsets of computer viruses
  of a type of computer viruses that are determined to be present.

CLMS(2)

 2. The method for increasing the speed at which a computer, which has
files including more than one fork, scans for the presence of a computer
virus of claim 1 further comprising the step of scanning files for which
said identifying information is not found in said scan information cache
for a subset of those viruses which infect said computer, said subset
including viruses that can infect said files.

CLMS(3)

 3. The method for increasing the speed at which a computer, which has
files including more than one fork, scans for the presence of a computer
```

virus of claim 1 further comprising the step of updating said scan information cache by placing a specific indicative value in some part of each location in said scan information cache which corresponds to a file in which a virus is found.

CLMS(4)

4. The method for increasing the speed at which a computer, which has files including more than one fork, scans for the presence of a computer virus of claim 1 further comprising the step of updating said scan information cache with new information concerning a state of a file for each file in which no virus is found.

CLMS(5)

5. The method for increasing the speed at which a computer, which has files including more than one fork, scans for the presence of a computer virus of claim 1 further comprising the step of updating said scan information cache with new information concerning a state of a file for each file in which a virus is found.

CLMS(6)

6. The method for increasing the speed at which a computer, which has files including more than one fork, scans for the presence of a computer virus of claim 1 further comprising the step of comparing the difference between said at least one length in said scan information cache for the initial state of said file and said at least one length of said current state of said file to a tolerance.

CLMS(7)

7. An apparatus that can rapidly scan for the presence of a computer virus on a computer, which has files including more than one fork, said apparatus comprising:
a scan information cache on a non-volatile storage medium;
means for gathering identifying information, which includes at least one length of some portion of a file, about an initial state of said file:
means for storing said identifying information in said scan information cache;
means for gathering state information, which includes at least one length of some portion of said file, about a current state of said file;
means for determining how said identifying information stored in the scan information cache differs from said current state information for said file thereby indicating a presence or absence of one or more subsets of computer viruses, said one or more subsets each including

one or more viruses which affect state information of said file in certain characteristic manners;

means for scanning said file for one or more of said subsets of computer viruses of a type of computer viruses that are determined to be present.

CLMS(8)

8. The apparatus that can rapidly scan for the presence of a computer virus of claim 7 further comprising means for scanning files for which said identifying information is not found in said scan information cache for a subset of those viruses which infect said computer, said subset including viruses that can infect said files.

CLMS(9)

9. The apparatus that can rapidly scan for the presence of a computer virus of claim 7 further comprising means for updating said scan information cache by placing a specific indicative value in some part of each location in said scan information cache which corresponds to a file in which a virus is found.

CLMS(10)

10. The apparatus that can rapidly scan for the presence of a computer virus of claim 7 further comprising means for updating said scan information cache with new information concerning a state of a file for each file in which no virus is found.

CLMS(11)

11. The apparatus that can rapidly scan for the presence of a computer virus of claim 7 further comprising means for updating said scan information cache with new information concerning a state of a file for each file in which a virus is found.

CLMS(12)

12. The apparatus that can rapidly scan for the presence of a computer virus of claim 7 further comprising means for comparing the difference between said at least one length in said scan information cache for the initial state of said file and said at least one length of said current state of said file to a tolerance.

=>

```
1.  **5,537,540**, Jul. 16, 1996, Transparent, secure computer virus
detection method and apparatus; Craig A. Miller, et al., 395/183.14,
183.12, 185.05 [IMAGE AVAILABLE]
=> display clms
ENTER (L7), L# OR ?:l7
ENTER ANSWER NUMBER OR RANGE (1):1

US PAT NO:     **5,537,540** [IMAGE AVAILABLE]         L7: 1 of 1

CLAIMS:

CLMS(1)

 What is claimed is:

 1. A method for operating a computer system, the computer system
including a processor; random access memory; read only memory containing
a ROM program executed by said processor upon resetting of the computer
system; at least one storage means having a non-DOS partition and at
least one other partition, said non-DOS partition having a first and
second region, said first region for storing a first verification program
executed by said processor, a first verification list for storing a list
of files stored on said second region, said files including files
required to boot the computer system and a second verification program,
and a first hash code table for storing hash codes of said first
verification list files, said second region for storing a first operating
system and the second verification program executed by said processor, a
second verification list for storing a list of files stored on said other
partitions and a second hash code table, said other partitions include at
least a second partition for storing a second operating system and user
programs executed by said processor; and a non-volatile memory having a
plurality of locations for storing an non-volatile memory hash code and
accessible to said processor, said non-volatile memory hash code
containing at least one value being a modification detection code of said
first region, said plurality of locations of said non-volatile memory
being readable and writable by said processor after a first reset of the
computer system, being write protected after receipt of a designated
signal from said processor, and being made writable again only after a
second reset of the computer system, the method comprising the steps of:
 resetting the computer system and executing said ROM program, whereupon
  the ROM program causes execution of the following steps:
  computing a hash code for said first region of said non-DOS partition;
  determining if said computed hash code is equal to said non-volatile
   memory hash code value stored in said non-volatile memory;
  loading said first verification program stored on said non-DOS
   partition into said random access memory if said computed hash code is
```

equal; and
executing said first verification program loaded into said random access memory; and
wherein said first verification program further causes execution of the following steps:
computing hash codes for files listed in said first verification list;
determining if said computed hash codes are equal to hash code values stored in said first hash code table; and
booting said first operating system on said non-DOS partition if said computed hash codes are equal; and
wherein said operating system further causes execution of the following steps upon booting:
loading said second verification program stored on said non-DOS partition into said random access memory; and
executing said second verification program loaded into said random access memory; and
wherein said second verification program further causes execution of the following steps:
computing hash codes for files listed in said second verification list;
determining if said computed hash codes are equal to ash code values stored in said second hash code table; and
returning control to said ROM program; and
whereupon the ROM program causes further execution of the following steps if said computed hash codes are equal:
providing said designated signal to said non-volatile memory device prior to booting said second operating system; and
booting said second operating system from said second partition.

CLMS(2)

2. The method of claim 1, wherein said storage means further includes a master boot record for containing a partition table, said non-DOS partition having a boot sector containing instructions for booting said first operating system and whereupon said first verification program further causes execution of the following steps prior to said booting said first operating system step:
computing hash codes for said master boot record and said non-DOS partition boot sector; and
determining if said hash codes are equal to hash code values stored in said first hash code table.

CLMS(3)

3. The method of claim 2, wherein the at least one other partition includes at least one DOS accessible partition, whereupon said second verification program further causes execution of the following steps prior to computing hash codes for files listed in said second

verification list:
determining each DOS accessible partition of said storage means;
assigning said non-DOS partition to a logical drive address value;
assigning a last partition of said DOS accessible partitions to a first logical drive address value after said non-DOS partition;
assigning a first partition of said DOS accessible partitions to a second logical drive address value after said non-DOS partition; and
assigning the remaining partitions of said DOS accessible partition with logical drive address values sequentially incremented from said second logical drive address.

CLMS(4)

4. The method of claim 2, wherein said non-DOS partition includes system files of said second operating system, and backups of said system files of said first and second operating systems, whereupon said first verification program further causes execution of the following steps prior to said booting said first operating system step:
determining if system files from said second operating system are present on said second partition;
verifying said system files of said second operating system stored on said non-DOS partition if said system files of said second operating system are present on said second partition; and
copying said system file backups of said first operating system stored on said non-DOS partition to said system files of said first operating system on said non-DOS partition if said system files of said second operating system are not present on said second partition.

CLMS(5)

5. The method of claim 4, whereupon said verifying step further causes execution of the following steps:
locating said system files of said second operating system stored on said non-DOS partition;
computing hash codes for said system files; and
determining if said computed hash codes are equal to hash code values stored in said first hash code table.

CLMS(6)

6. The method of claim 4, whereupon said verifying step further causes execution of the following step prior to copying said system file backups:
verifying said system file backups of said second operating system stored on said non-DOS partition if said hash codes are not equal.

CLMS(7)

7. The method of claim 6, whereupon said verifying system file backups step further causes execution of the following steps:
computing hash codes for said system file backups of said second operating system stored on said non-DOS partition;
determining if said computed hash codes are equal to hash code values stored in said first hash code table; and
restoring said system files of said second operating system to said non-DOS partition from an external source if said hash codes are not equal.

CLMS(8)

8. The method of claim 4, wherein said second partition includes a storage compression utility, and whereupon said first verification program further causes execution of the following steps prior to said booting said first operating system step:
copying said storage compression utility from said second partition to said non-DOS partition;
computing a hash code for said storage compression utility;
determining if said computed hash code is equal to a hash code value stored in said first hash code table; and
deleting said storage compression utility copy from said non-DOS partition if said hash codes are not equal.

CLMS(9)

9. The method of claim 8, wherein said second partition includes a boot record for booting said second operating system, whereupon said second verification program further causes execution of the following steps prior to said returning control to said ROM program step:
computing a hash code for said second partition boot record; and
determining if said computed hash code is equal to a hash code value stored in said second hash code table.

CLMS(10)

10. The method of claim 1, wherein said first verification program further causes execution of the following step prior to computing hash code values for files listed in said first verification list:
saving a lower region of said random access memory, said lower region containing ROM status information; and
wherein said second verification program further causes execution of the following step just prior to returning control to said ROM program:
restoring said lower region of said random access memory.

CLMS(11)

```
 11. The method of claim 1, wherein said first and second operating
systems are DOS.

CLMS(12)

 12. The method of claim 11, wherein said second verification list
includes a system configuration files portion and a user files portion.

CLMS(13)

 13. The method of claim 1, wherein said first operating system is DOS
and said second operating system is other than DOS.

CLMS(14)

 14. The method of claim 13, wherein said second verification list
includes only system configuration files.
=>
```

```
1.  **5,473,769**, Dec. 5, 1995, Method and apparatus for increasing the
speed of the detecting of computer viruses; Paul D. Cozza, 395/183.15;
380/4; 395/440 [IMAGE AVAILABLE]
=> display clms
ENTER (L8), L# OR ?:18
ENTER ANSWER NUMBER OR RANGE (1):1

US PAT NO:     **5,473,769** [IMAGE AVAILABLE]          L8: 1 of 1

CLAIMS:

CLMS(1)

 What is claimed is:

 1. A method for increasing the speed at which a computer scans for the
presence of a computer virus, said method comprising the steps of:
 gathering initial state information about an initial state of a file and
  storing said initial state information in a scan information cache on a
  non-volatile storage medium;
 storing said initial state information in said scan information cache;
 gathering current state information about a current state of said file;
 determining how said initial state information stored in said scan
  information cache for said file differs from said current state
  information thereby indicating a presence or absence of one or more
  subsets of computer viruses, said one or more subsets each including
  one or more viruses which affect state information of said file in
  certain characteristic manners;
 scanning said file for one or more of said subsets of computer viruses
  of a type of computer viruses that are determined to be present.

CLMS(2)

 2. The method for increasing the speed at which a computer scans for the
presence of a computer virus of claim 1 further comprising the step of
scanning said file, for which said initial state information is not found
in said scan information cache, for a subset of those viruses which
infect computers of the same type as said computer, said subset including
viruses that can infect files of the same type as said file.

CLMS(3)

 3. The method for increasing the speed at which a computer scans for the
presence of a computer virus of claim 1 further comprising the step of
updating said scan information cache by placing a specific indicative
value in at least one memory location in said scan information cache,
```

said at least one memory location corresponding to a file in which a virus is found.

CLMS(4)

4. The method for increasing the speed at which a computer scans for the presence of a computer virus of claim 1 further comprising the step of updating said scan information cache with new information concerning a state of said file if no virus is found in said file.

CLMS(5)

5. A method for increasing the speed at which a computer scans for the presence of a computer virus, said method comprising the steps of:
gathering initial state information about an initial state of a volume and storing said initial state information in a scan information cache on a non-volatile storage medium;
storing said initial state information in said scan information cache;
gathering current state information about a current state of said volume;
determining how said initial state information stored in said scan information cache for said volume differs from said current state information for said volume thereby indicating a presence or absence of one or more subsets of computer viruses, said one or more subsets each including one or more viruses which affects state information of said volume in certain characteristic manners;
scanning said volume for one or more of said subsets of computer viruses of a type of computer viruses that are determined to be present.

CLMS(6)

6. The method for increasing the speed at which a computer scans for the presence of a computer virus of claim 5 further comprising the step of scanning said volume, for which said initial state information is not found in said scan information cache, for a subset of those viruses which infect computers of the same type as said computer, said subset including viruses that can infect volumes of the same type as said volume.

CLMS(7)

7. The method for increasing the speed at which a computer scans for the presence of a computer virus of claim 5 further comprising the step of updating said scan information cache by placing a specific indicative value in at least one memory location in said scan information cache, said at least one memory location corresponding to a volume in which a virus is found.

CLMS(8)

8. The method for increasing the speed at which a computer scans for the presence of a computer virus of claim 5 further comprising the step of updating said scan information cache with new information concerning a state of said volume if no virus is found in said volume.

CLMS(9)

9. An apparatus that can rapidly scan for the presence of a computer virus comprising:
a scan information cache on a non-volatile storage medium;
means for gathering initial state information about an initial state of a file stored on a computer storage medium and storing said initial state information in said scan information cache;
means for storing said initial state information in said scan information cache;
means for gathering current state information about a current state of said file;
means for determining how said initial state information stored in said scan information cache for said file differs from said current state information for said file thereby indicating a presence or absence of one or more subsets of computer viruses, said one or more subsets each including one or more viruses which affect state information of said file in certain characteristic manners;
means for scanning said file for one or more of said subsets of computer viruses of a type of computer viruses that are determined to be present, said means for scanning being operated by a computer having access to said computer storage medium.

CLMS(10)

10. The apparatus that can rapidly scan for the presence of a computer virus of claim 9 further comprising means for scanning said file, for which said initial state information is not found in said scan information cache, for a subset of those viruses which infect computers of the same type as said computer, said subset including viruses that can infect files of the same type as said file.

CLMS(11)

11. The apparatus that can rapidly scan for the presence of a computer virus of claim 9 further comprising means for updating said scan information cache by placing a specific indicative value in at least one memory location in said scan information cache, said at least one memory location corresponding to a file in which a virus is found.

CLMS(12)

12. The apparatus that can rapidly scan for the presence of a computer virus of claim 9 further comprising means for updating said scan information cache with new information concerning a state of said file if no virus is found in said file.

CLMS(13)

13. An apparatus that can rapidly scan for the presence of a computer virus comprising:
a scan information cache on a non-volatile storage medium;
means for gathering initial state information about an initial state of a volume stored on a computer storage medium and storing said initial state information in said scan information cache;
means for storing said initial state information in said scan information cache;
means for gathering current state information about a current state of said volume;
means for determining how said initial state information stored in said scan information cache for said volume differs from said current state information for said volume thereby indicating a presence or absence of one or more subsets of computer viruses, said one or more subsets each including one or more viruses each which affect state information of said volume in certain characteristic manners;
means for scanning said volume for one or more of said subsets of computer viruses of a type of computer viruses that are determined to be present, said means for scanning being operated by a computer having access to said computer storage medium.

CLMS(14)

14. The apparatus that can rapidly scan for the presence of a computer virus of claim 13 further comprising means for scanning said volume, for which said initial state information is not found in said scan information cache, for a subset of those viruses which infect computers of the same type as said computer, said subset including viruses that can infect volumes of the same type as said volume.

CLMS(15)

15. The apparatus that can rapidly scan for the presence of a computer virus of claim 13 further comprising the step of updating said scan information cache by placing a specific indicative value in at least one memory location in said scan information cache, said at least one memory location corresponding to a volume in which a virus is found.

```
CLMS(16)

 16. The apparatus that can rapidly scan for the presence of a computer
virus of claim 13 further comprising the step of updating said scan
information cache with new information concerning a state of said volume
if no virus is found in said volume.
=>
```

```
  **5,408,642**, Apr. 18, 1995, Method for recovery of a computer
program infected by a computer virus; Omri Mann, 395/183.14; 371/30, 67.1
[IMAGE AVAILABLE]
=> display clms
ENTER (L9), L# OR ?:l9
ENTER ANSWER NUMBER OR RANGE (1):1

US PAT NO:     **5,408,642** [IMAGE AVAILABLE]         L9: 1 of 1

CLAIMS:

CLMS(1)

 I claim:

 1. A method of restoring a computer program infected by a computer virus
by removing said virus comprising:
 a) a generating a unique fingerprint for a computer program prior to
  said program being infected;
 b) prior to said program being infected storing said unique fingerprint
  and data relating to a beginning portion of said program at a separate
  location;
 c) utilizing said stored data for generating a fingerprint of a string
  in said infected program;
 d) comparing the fingerprint generated in step c) with the fingerprint
  stored in step b) and determining that the program can be restored if
  said fingerprints match;
 e) restoring said program from said string and said stored data if said
  comparison in step d) determines that restoration is possible.

CLMS(2)

 2. The method according to claim 1 further comprising the steps of:
 f) selecting a string adjacent said string;
 g) repeating steps c), d) and e) of claim 1;
 h) repeating steps f) and g) until said program is restored or a
  predetermined number of strings have been tested.

CLMS(3)

 3. The method according to claim 1 wherein said string is a
concatenation of a plurality of sub-strings.

CLMS(4)

 4. The method according to claim 1 wherein said data stored in step b)
includes a length of said program and said string utilized in step c) is
```

of said length.

CLMS(5)

5. The method according to claim 1 wherein said beginning portion of said program includes a header of the program.

CLMS(6)

6. The method according to claim 1 wherein said fingerprint comprises an error detecting code for said program.

CLMS(7)

7. The method according to claim 6 wherein said error detecting code is a cyclic redundancy check.

CLMS(8)

8. The method according to claim 6 wherein said error detecting code is a checksum.

CLMS(9)

9. The method according to claim 6 wherein said error detecting code is an exclusive - OR of words in said program.

CLMS(10)

10. A method of restoring a computer program infected by a computer virus by removing said virus comprising:
a) generating a unique fingerprint for a computer program prior to said program being infected;
b) storing, at a separate location, said unique fingerprint, data relating to a beginning portion of said program and a length of said program prior to said program being infected;
c) selecting a starting location in said program having an address which exceeds a length of said stored beginning portion by a predetermined number;
d) generating a fingerprint of a continuous string in said program, having a length equal to said stored program length, utilizing said stored data relating to said beginning portion;
e) comparing the fingerprint generated in step c) with the fingerprint stored-in step b) and determining that the program can be restored if said fingerprints match;
f) restoring said program from said string and said stored data if said comparison in step e) determines that restoration is possible.

CLMS(11)

11. The method according to claim 10 further comprising the steps of:
g) incrementing the address of said starting location;
h) repeating steps d), e) and f);
i) repeating steps g) and h) until said program is restored or a predetermined number of increments in said address have been implemented.

CLMS(12)

12. The method according to claim 10 wherein said fingerprint comprises an error detecting code for said program.

CLMS(13)

13. The method according to claim 12 wherein said error detecting code is a cyclic redundancy check.

CLMS(14)

14. The method according to claim 12 wherein said error detecting code is a checksum.

CLMS(15)

15. The method according to claim 12 wherein said error detecting code is an exclusive - OR of words in said program.

CLMS(16)

16. A method of restoring a computer program infected by a computer virus by removing said virus comprising:
a) generating a unique fingerprint for a computer program prior to said program being infected;
b) storing, at a separate location, said unique fingerprint, and a length of said program prior to said program being infected;
c) generating a fingerprint of a string formed of first and second sub-strings, said first sub-string having a starting point which is after an ending point of said program and having a predetermined length, said second sub-string having a starting point offset from a beginning of said program by said predetermined length, the combined lengths of said first and second sub-strings being equal to said stored program length, said fingerprint being generated utilizing said stored data relating to said beginning portion;
d) comparing the fingerprint generated in step c) with the fingerprint

stored in step b) and determining that the program can be restored if said fingerprints match;
e) restoring said program from said string and said stored data if said comparison in step d) determines that restoration is possible.

CLMS(17)

17. The method according to claim 16 further comprising the steps of:
f) incrementing said predetermined length;
g) repeating steps c), d) and e);
h) repeating steps f) and g) until said program is restored or a predetermined number of increments in said starting point has been implemented.

CLMS(18)

18. The method according to claim 16 wherein said fingerprint comprises an error detecting code for said program.

CLMS(19)

19. A method of restoring a computer program having fixup tables and being infected by a computer virus by removing said virus comprising:
a) generating a unique fingerprint for a computer program prior to said program being infected;
b) storing, at a separate location, said unique fingerprint, data relating to a beginning portion, of said program and a length of said program prior to said program being infected;
c) determining a length of said virus;
d) determining an integer fixup shift equal to said virus length divided by 16;
e) generating a modified fixup table having a segment portion of each address reduced by said fixup shift;
f) generating a fingerprint of a concatenation of said stored beginning portion, said modified fixup table and a continuous string having a starting point equal to 16 times said fixup shift bytes after an end of said fixup table and having a length of said stored length minus a length of said fixup table and minus said stored length of said beginning portion;
g) comparing the fingerprint generated in step c) with the fingerprint stored in step b) and determining that the program can be restored if said fingerprints match;
h) restoring said program from said string and said stored data if said comparison in step d) determines that restoration is possible.

CLMS(20)

20. The method according to claim 19 further comprising the steps of:
i) decrementing said integer fixup shift;
j) repeating steps d), e), f), g) and h)
k) repeating steps i) and j) until said program is restored or a predetermined number of decrements of said fixup shift have been implemented.

CLMS(21)

21. The method according to claim 19 wherein said fingerprint comprises an error detecting code for said program.
=>